# 实验六　寄存器

**2020 年秋季学期**

杨飞洋 191220138

***目录***

*1. 实验目的*

*2. 实验原理*

*3. 实验环境/器材*

*4. 设计思路*

*5. 实验步骤*

*6. 测试方法*

*7. 实验结果*

*8. 实验中遇到的问题*

*9. 实验得到的启示*

*10. 意见和建议*

*11. 思考题*

## 1. 实验目的：

学习常用寄存器的设计方法，复习寄存器的原理，学习寄存器和常用的移位寄存器的设计。利用移位寄存器实现随机数发生器。

## 2. 实验原理（知识背景）

寄存器也是最常用的时序逻辑电路，是一种存储电路。寄存器的存储电路 是由锁存器或触发器构成的，因为一个锁存器或触发器能存储 1 位二进制数，所以由 N 个锁存器或触发器可以构成 N 位寄存器。

## 3. 实验环境/器材

系统：windows10
开发软件：Quartus 17.1 Lite
开发板：DE10 Standard
仿真环境：ModelSim
芯片：Cyclone V ， 5CSXFC6D6

## 4.设计思路

*第一部分：*

一共有一个输入的时钟信号，一个三位的二进制选择端 chos，一个 8 位输入用于置数，一个 left 位用于左端串行输入，还有一个输出的 8 位二进制。

```
module shifter(
    input clk,
    input [2:0]chos,
    input [7:0]inp,
    input left,
    output reg [7:0]result
);
```

一共要实现 8 个功能，总的用一个 case 语句

```
case(chos)
0:result <= 0;
1:result <= inp;
2:result <= (result>>1);
3:result <= (result<<1);
4:result <= {result[7],result[7:1]};
5:result <= {left,result[7:1]};
6:result <= {result[0],result[7:1]};
7:result <= {result[6:0],result[7]};
endcase
```

0：清零

1：置数
2：逻辑右移
3：逻辑左移
4：算术右移
5：串行输入
6：循环右移
7：循环左移

*第二部分：*

用一个右移的移位寄存器来实现随机数发生器，反馈方程为
r_in = x[0]^[2]^x[3]^x[4]，如果出现了 0，就随便赋一个值，不然一直是零下去了。
代码为：

```
module randint
#(
    parameter n
)
(
    input clk,
    input reset,
    input [7:0]start,
    output reg [7:0]r = 0,
    output [6:0]ten,
    output [6:0]one
);
    integer i;
    reg x;


always @(posedge clk)begin
    if(reset)
       r <= start;
    else if(r == 0)
       r <= 8'b01100110;
    else begin
       x = 0;
       for(i = 0;i<8;i = i+1)
          x = x^(n[i]&r[i]);
       r <= {x,r[7:1]};
    end
end
```

参数 n 是为了可以进行扩展，因为 N 位的随机数发生器都可以用异或运算来解决，这里的 n 就可以赋为 8'b00011101.
再用一个分频器输出一个周期 1s 的信号，用 7 段数码管解释器输出数字

分频器模块：

```
module divider
#(
    parameter n
)
(
    input clk,
    output reg my_signal = 0
);
    integer clk_count = 0;

    always @ (posedge clk) begin
       if(clk_count == 25000000*n) begin
          clk_count <= 0;
          my_signal <= ~my_signal;
       end
       else
          clk_count <= clk_count + 1;
    end

endmodule
```

数字解释模块：

```
module trsltr(number,s);
    input [3:0]number;
    output reg [6:0]s;

    always @(*)
       case(number)
          0:s = 7'b1000000;
          1:s = 7'b1111001;
          2:s = 7'b0100100;
          3:s = 7'b0110000;
          4:s = 7'b0011001;
          5:s = 7'b0010010;
          6:s = 7'b0000010;
          7:s = 7'b1111000;
          8:s = 7'b0000000;
          9:s = 7'b0010000;
          10:s = 7'b0001000;
          11:s = 7'b0000011;
          12:s = 7'b1000110;
          13:s = 7'b0100001;
          14:s = 7'b0000110;
          15:s = 7'b0001110;
          default:s = 7'b1111111;
       endcase

endmodule
```

顶层文件中的调用：

```
wire signal;
divider#(1) d(CLOCK_50,signal);

randint#(8'b00011101) R(signal,SW[8],SW[7:0],LEDR[7:0],HEX1[6:0],HEX0[6:0]);
```

## 5.实验步骤 （第一部分和第二部分类似）

### 一．利用 systembuilder 建立工程

选择 CLOCK、SWITCH、LEDR、button

### 二．编写 Verilog 代码

写一下相应模块的代码，在顶层文件中做相应的调用，进行编译调试，直至成功。

### 三．在板子上进行测试

## 6.测试方法

在板子上进行各个功能的测试，随机数记录一下出现的数字，做到不重复。

## 7.实验结果

已验收。

## 8.实验中遇到的问题

1.对于 LFSR 的形成原理有点不懂，最后就用了书本给的反馈方程。
2.一开始用 switch 键模拟时钟信号，会出现异常，是因为 switch 没有消抖，所以用 key 键模拟时钟信号。

## 9.实验得到的启示

1．一定要学会用分模块编程，会让结构更加清晰，debug 更加容易
2．有问题可以向老师助教请教、和同学讨论、上网搜，很多时候是思路或结构出现了问题。

## 10.意见和建议

1.这种实验测试文件不好测试，希望老师可以直接说清楚不用测试文件，只需要在板子上测试。
2.希望给一实验原理的指导，不然自己搞有点懵。

## *11.思考题*

我觉得不用拘泥于用一位移位寄存器来做随机数发生器，这样感觉有点死板，而且可能会有一定规律。不妨就每一位都对应一个反馈方程，反馈方程中需要异或运算，因为异或是更加随机的。虽然这个方法可能不能保证遍历每一个值，不过可以用计算机来得出这样一组反馈方程，会是更加复杂的一组伪随机数序列。